

同步异步

练习1

```
1 <script>
2     console.log(1, 'script start') // 1
3     setTimeout(function () {
4         console.log(3, 'settimeout') // 3
5     })
6     console.log(2, 'script end') // 2
7 </script>
```

练习2

```
1 console.log(1, 'script start');//
2
3     let promise1 = new Promise(function (resolve) {
4         console.log(2, 'promise1');//
5         resolve();
6         console.log(3, 'promise1 end');//
7     }).then(function () {
8         console.log(5, 'promise2');//
9     });
10
11     setTimeout(function () {
12         console.log(6, 'settimeout');//
13     });
14
15     console.log(4, 'script end');//
```

练习3

```
1 async function async1() {
2     console.log(2, 'async1 start');//
3     await async2();
4     console.log(5, 'async1 end');//
5 }
6
```

```
7     async function async2() {
8         console.log(3, 'async2');//
9     }
10
11     console.log(1, 'script start');//
12
13     async1();
14
15     console.log(4, 'script end');//
```

练习4

```
1  async function async1() {
2      console.log(2, 'async1 start'); //2
3      await async2();
4      console.log(6, 'async1 end'); //6
5  }
6
7  async function async2() {
8      console.log(3, 'async2'); //3
9  }
10
11  console.log(1, 'script start'); //1
12
13  setTimeout(function () {
14      console.log(8, 'setTimeout'); //8
15  }, 0)
16
17  async1();
18
19  new Promise(function (resolve) {
20      console.log(4, 'promise1'); //4
21      resolve();
22  }).then(function () {
23      console.log(7, 'promise2'); //7
24  });
25
26  console.log(5, 'script end'); //5
```

面试题:

题1

```
1 //给函数参数设置默认值的时候，需要设置到尾参数身上
2 //参数是没有顺序要求，在函数内部任何地方都可以使用
3 //形参是函数内部的全局变量
4 function fn(b, c, a = 10) {
5     console.log(b, c, a);
6     return b + a + c;
7 }
8 fn(2, 3)
```

题2

```
1 function fun(n, o) {
2     console.log(o);
3     return {
4         fun: function (m) {
5             return fun(m, n);
6         }
7     };
8 }
9
10 // 1-----
11 var a = fun(0); //1 undefined ---->形参没有接受到实参
12 /*****
13 分析: AO--> n:0
14       a: {
15           fun: function(m) {
16               return fun (m, n);
17           }
18       }
19 */
20 a.fun(1); //2 o-->n-->0
21 /** 分析:
22       1-->m ---->return fun(1, 0);
23 */
```

```

24     a.fun(2); //3 o--->n--->0
25     /** 分析:
26         2-->m --->return fun(2, 0);
27     */
28     a.fun(3); //4 o--->n--->0
29     a.fun(32227878787882);
30
31     // 2-----
32     var b = fun(0).fun(1).fun(2).fun(3).fun(500).fun(0); //undefined-->0-->1-->2--
>500
33     /*
34     分析: fun(0): A0--> n:0
35         b: {
36             fun: function(m) {
37                 return fun (m, n);
38             }
39         }
40     */
41
42     // 3-----
43     var c = fun(0).fun(1); //undefined 0
44     /**
45     c 这个对象里面的n-->1
46     c: {
47         fun: function (m) {
48             return fun(m, n);
49         }
50     }
51     */
52     c.fun(2); // 1
53     c = c.fun(3); //1-->3
54     c.fun(999); //3
55     c.fun(123); //3

```

题3: 数组去重

```

1     let arr1 = [1, 2, 3, 2, 1];
2
3     //1

```

```
4     function p1(arr) {
5         let newArr = [];
6         for (let i = 0; i < arr.length; i++) {
7             newArr.indexOf(arr[i]) === -1 && newArr.push(arr[i]);
8         }
9         return newArr;
10    }
11
12    //2
13    function p2(arr) {
14        return Array.from(new Set(arr));
15    }
16
17    //3
18    function p3(arr) {
19        return [...new Set(arr)];
20    }
21
22    //4
23    function p4(arr) {
24        return arr.filter((item, ind) => arr.indexOf(item) === ind);
25    }
26
27    console.log(p1(arr1)); //[1, 2, 3]
28    console.log(p2(arr1)); //[1, 2, 3]
29    console.log(p3(arr1)); //[1, 2, 3]
30    console.log(p4(arr1)); //[1, 2, 3]
31
32    //reduce
33    var arr = [1, 2, 3, 4, 5, 6, 4, 3, 8, 1];
34    // 数组去重:
35    // 方法12 : reduce
36    function newArrFn(arr) {
37        let newArr = [];
38        return arr.reduce((prev, next, index, arr) => {
39            // 如果包含, 就返回原数据, 不包含, 就把新数据追加进去
40            return newArr.includes(next) ? newArr : newArr.push(next);
41        }, 0)
42    }
43    console.log(newArrFn(arr));
```

```

44
45 //数组的indexOf方法：遍历数组，找到第一个传入的参数，如果没有就返回-1
46 let arr = [1, 2, 2, 2, 34, 4, 34, 5];
47
48 function fn(arr) {
49     let newArr = [];
50     for (let i = 0; i < arr.length; i++) {
51         if (newArr.indexOf(arr[i]) === -1) {
52             newArr.push(arr[i]);
53         }
54     }
55     return newArr;
56 }
57 console.log(fn(arr)); //[1, 2, 34, 4, 5]
58
59 //定义一个新数组，放入原数组第一项，双重for循环遍历对比每一项，没有重复的就加入新数
组：
60 let arr = [1, 2, 2, 2, 343, 4, 34, 5];
61
62 function fn(arr) {
63     let newArr = [arr[0]];
64     for (let i = 0; i < arr.length; i++) {
65         let flag = false;
66         for (let j = 0; j < newArr.length; j++) {
67             if (arr[i] === newArr[j]) {
68                 flag = true;
69                 break;
70             }
71         }
72         if (!flag) {
73             newArr.push(arr[i]);
74         }
75     }
76     return newArr;
77 }
78 console.log(fn(arr)); //[1, 2, 343, 4, 34, 5]
79
80 //数组的includes方法用于判断判断一个数组是否包含一个指定的值。如果是则返回 true，否则
返回 false。
81 let arr = [1, 2, 2, 2, 343, 4, 34, 5];

```

```

82
83     function fn(arr) {
84         let newArr = [];
85         for (let i = 0; i < arr.length; i++) {
86             if (!newArr.includes(arr[i])) {
87                 newArr.push(arr[i]);
88             }
89         }
90         return newArr;
91     }
92     console.log(fn(arr)); //[1, 2, 343, 4, 34, 5]

```

93
94 //filter方法是根据过滤条件对整个数组进行过滤，符合的为true保留，不符合的为false删除。

```

95     let arr = [1, 2, 2, 2, 343, 4, 34, 5];
96
97     let arr2 = arr.filter(function (item, ind) {
98         return arr.indexOf(item) === ind;
99     });
100     console.log(arr2);

```

101
102 //ES6的set方法去重

```

103     let arr = [1, 2, 2, 2, 3, 4, 34, 5];
104     let arr1 = [1, 2, , 2, 4, 34, 5]

```

```

105
106     let arr2 = [...new Set(arr)];
107     let arr3 = [...new Set(arr, arr1)];

```

```

108
109     console.log(arr2); // [1, 2, 343, 4, 34, 5]
110     console.log(arr3);

```

111
112 //当数组里含有复杂数据类型的时候，一般的去重方法就没用了，
113 //但是可以用filter和hasOwnProperty方法去重

```

114     let arr = [1, 1, 2, 2, 3, 3, {}, {}, null, null, NaN, NaN, [],
115         [], '', '', false, false, true, true
116     ];
117     let arr1 = [];
118     arr1 = arr.filter(function (item) {
119         if (!arr.hasOwnProperty(typeof item + item)) {
120             return arr[typeof item + item] = true;

```

```

121     }
122 });
123 console.log(arr1);
124 //[1, 2, 3, {}, null, NaN, [], '', false, true]
125
126 //遍历数组，利用数组的indexOf方法判断此元素在该数组中首次出现的索引
127 //与循环的i是否相等;再利用splice方法删除不符合条件的元素,并及时调整i;
128 let arr = [1, 2, 3, 4, 5, 3, 2, 3, 4, 1, 11, 22, 2, 5, 6];
129
130 function myArr(arr) {
131     for (let i = 0; i < arr.length; i++) {
132         if (arr.indexOf(arr[i]) !== i) {
133             arr.splice(i, 1);
134             i--;
135         }
136     }
137     return arr;
138 }
139 console.log(myArr(arr));
140 //[1, 2, 3, 4, 5, 11, 22, 6]
141
142 //主要利用findIndex 的特性，查找元素找不到就返回-1， 接下来
143 //就需要判断，如果是-1，说明没找到，就往新数组里面添加元素。
144 let arr = [1, 2, 3, 4, 5, 3, 2, 3, 4, 1, 11, 22, 2, 5, 6];
145
146 function myArr(arr) {
147     let newArr = [];
148     for (let i = 0; i < arr.length; i++) {
149         newArr.indexOf(arr[i]) === -1 ? newArr.push(arr[i]) : newArr;
150     }
151     return newArr;
152 }
153 console.log(myArr(arr));
154 //[1, 2, 3, 4, 5, 11, 22, 6]
155
156 //首先利用 sort 方法进行排序。进行循环，如果原数组的第 i 项和新数组的 i - 1 项不一致，就push进去。
157 let arr = [1, 2, 3, 4, 5, 3, 2, 3, 4, 1, 11, 22, 2, 5, 6];
158
159 function myArr(arr) {

```



```

160         arr = arr.sort();
161         let newArr = [];
162         for (let i = 0; i < arr.length; i++) {
163             arr[i] === arr[i - 1] ? newArr : newArr.push(arr[i]);
164
165         }
166         return newArr;
167     }
168     console.log(myArr(arr)); //[1, 11, 2, 22, 3, 4, 5, 6]
169
170     //利用 set数据不重复的特点, 结合 Array.from
171     let arr = [1, 2, 3, 4, 5, 3, 2, 3, 4, 1, 11, 22, 2, 5, 6];
172
173     function newArrFn(arr) {
174         // .new Set方法, 返回是一个类数组, 需要结合 Array.from , 转成真实数组
175         return (Array.from(new Set(arr)))
176     }
177     console.log(newArrFn(arr));
178
179     //利用数据结构存值的特点
180     var arr = [1, 2, 3, 4, 5, 6, 4, 3, 8, 1]
181     // 数组去重:
182     // 方法11 : Map
183     function newArrFn(arr) {
184         let newArr = []
185         let map = new Map()
186         for (let i = 0; i < arr.length; i++) {
187             // 如果 map里面不包含, 就设置进去
188             if (!map.has(arr[i])) {
189                 map.set(arr[i], true)
190                 newArr.push(arr[i])
191             }
192         };
193         return newArr
194     }
195     console.log(newArrFn(arr));

```

题4: 数组拷贝

```
1  let arr1 = [1, 2, 3, {
2      a: 1,
3      b: [3, 4, 5]
4  }, undefined, [5, 6]];
5  /*
6  function copyArray(arr) {
7      let newArray = [];
8      //循环
9      for (let j = 0; j < arr.length; j++)newArray.push(arr[j]);
10     return newArray;
11 }
12
13 function copyArray(arr) {
14     return [...arr];
15 }
16
17 function copyArray(arr) {
18     return JSON.parse(JSON.stringify(arr));
19 }
20 */
21
22 //浅拷贝-->深拷贝
23 let arr2 = [1, 2, 3, {
24     a: 1,
25     b: [3, 4, 5]
26 }, undefined, [5, 6]];
27
28 function copyArray(arr) {
29     if (arr instanceof Array) {
30         let newArray = [];
31         for (let j = 0; j < arr.length; j++) newArray.push(copyArray(arr[j]));
32         return newArray;
33     } else if (arr instanceof Object) {
34         let buffer = {};
35         for (let k in arr) buffer[k] = copyArray(arr[k]);
36         return buffer;
37     } else {
38         //找到边界墙 临界点
39         return arr;
```

```

40     }
41 }
42 console.log(copyArray(arr1), copyArray(arr1) === arr1);
43 console.log(copyArray(arr1)[5] === arr1[5]); //false
44 console.log(copyArray(arr1)[3].b === arr1[3].b); //false
45 console.log(copyArray(arr2));//[1, 2, 3, {...}, undefined, Array(2)]

```

题5: 回文

```

1 let str1 = '121';
2 function reverStr(str) {
3     str = str.toString();
4     return str === str.split('').reverse().join('');
5 }
6 console.log(reverStr(str1));
7 console.log(reverStr('str1'));
8 console.log(reverStr('666'));

```

题6: 函数调用

```

1 function Foo() {
2     getName = function () {
3         console.log(1);
4     };
5     console.log(this);
6     return this;
7 }
8
9 let getName = function () {
10     console.log(4);
11 };
12
13 Foo();
14 getName(); //1
15 console.log(Foo().getName);
16 Foo().getName();

```

▶ Window {window: Window, self: Window, document: document, name: '', location: Location, ...}	06.27day.面试题2.6.html:17
1	06.27day.面试题2.6.html:15
▶ Window {window: Window, self: Window, document: document, name: '', location: Location, ...}	06.27day.面试题2.6.html:17
undefined	06.27day.面试题2.6.html:27
▶ Window {window: Window, self: Window, document: document, name: '', location: Location, ...}	06.27day.面试题2.6.html:17
✖ ▶ Uncaught TypeError: Foo(...).getName is not a function at 06.27day.面试题2.6.html:28:15	06.27day.面试题2.6.html:28
>	

```

1  var getName = function () {
2      console.log(4);
3  };
4
5  function getName() {
6      console.log(5);
7  }
8  getName(); //4

```

题7：原型链查找

```

1  function foo() {
2      //直接修改静态方法
3      foo.a = function () { console.log(1) };
4      //给实例化追加一个方法 a
5      this.a = function () { console.log(2) };
6      //局部变量
7      a = function () { console.log(3) };
8      var a = function () { console.log(4) };
9  }
10  foo.prototype.a = function () { console.log(5) };
11  foo.a = function () { console.log(6) };//class static 静态方法
12  foo.a(); //6
13
14  var obj = new foo();//构造函数里面的代码是从头开始执行
15  console.log(obj);
16  //优先找自己，如果没有找到就顺着原型一层一层往上找，直到找到为止
17  obj.a(); //2
18  foo.a(); //1

```

题8：统计字符出现次数

```

1  // 'use strict';
2      var str = 'asdfs8scb8sasaa';
3      var json = {};
4      for (var i = 0; i < str.length; i++) {
5          if (!json[str.charAt(i)]) {
6              json[str.charAt(i)] = 1;
7          } else {
8              json[str.charAt(i)]++;
9          }
10     };
11
12     var iMax = 0;
13     var iIndex = '';
14     for (var i in json) {
15         if (json[i] > iMax) {
16             iMax = json[i];
17             iIndex = i;
18         }
19     }
20     console.log('出现次数最多的是:' + iIndex + '出现' + iMax + '次');

```

题9：字符串第二个单词以后首字母大写

```

1  // 'hua-qing-yuan-jian'
2      function result(str) {
3          var str1 = '';
4
5          str1 = str.toLowerCase().split('-');
6
7          //方法1
8          for (var i = 1; i < str1.length; i++) {
9              str1[i] = str1[i].charAt(0).toUpperCase() + str1[i].substring(1);
10         }
11
12         //方法2
13         /**
14         for (var j = 1; j < str1.length; j++) {

```

```

15         str1[j] = str1[j].substr(0, 1).toUpperCase() + str1[j].substr(1);
16     }
17     */
18
19     //方法3
20     /**
21     for (var n = 1; n < str1.length; n++) {
22         str1[n] = str1[n].substring(1, 0).toUpperCase() + str1[n].substring(1);
23     }
24     */
25     return str1.join('');
26 }
27
28 var str = 'hua-qing-yuan-jian';
29 console.log(result(str));

```

题10：找最大值

```

1     var arr = [-1, -2, 1, 10, 4, 5, 8];
2     var max1 = Math.max.apply(null, arr);
3     console.log(max1);
4
5     var max2 = arr.sort(function (a, b) {
6         return b - a;
7     })[0];
8     console.log(max2);
9
10    var max3 = -Infinity;
11    for (var i = 0; i < arr.length; i++) {
12        if (max3 < arr[i]) {
13            max3 = arr[i];
14        }
15    }
16    console.log(max3); //10

```

题12：获取时间及未来一周的时间

```

1      var d = new Date();
2      var today = d.getFullYear() + '-' + (d.getMonth() + 1).toString().padStart(2,
'0') + '-' + d.getDate();
3      console.log(today); //2022-06-12
4
5      d.setTime(d.getTime() + 7 * 24 * 3600 * 1000);
6      var nexttoday = d.getFullYear() + '-' + (d.getMonth() +
1).toString().padStart(2, '0') + '-' + d.getDate();
7      console.log(nexttoday); //2022-06-19 一周后的时间
8
9      var str = 'abcabcabcabcabda';
10     var arr = [];
11     var n = 0;
12     while (str.indexOf('ab', n) !== -1 && n < str.length) {
13         arr.push(str.indexOf('ab', n));
14         n = str.indexOf('ab', n) + 2;
15     }
16     console.log(arr);
17     console.log(n);

```

题13: sum求和1

```

1  function sum(...args) { //这里的三个点...是扩展运算符，该运算符将一个数组，变为参数序列。
2      if ([...args].length === 1) { //判断参数个数的形式是否为1个，即第二种形式
3          var cache = [...args][0]; //将第一个参数的值暂存在cache中
4          var add = function (y) { //创建一个方法用于实现第二个条件，最后并返回这个方法
5              cache += y;
6              console.log(cache)
7              return add;
8          }
9          return add;
10     } else { //这里就是参数的第一种形式
11         var res = 0; //这里最好先声明要输出的变量，并给其赋值，不然值定义而不赋值会输出NaN，因为js将undefined+number两个数据类型相加结果为NaN
12         for (var i = 0; i < [...args].length; i++) {
13             res += [...args][i]; //参数累加
14         }

```

```

15         console.log(res) //输出最后的累加结果
16     }
17 }
18 sum(2, 3, 4); //9
19 sum(2)(3)(4)(5); //5//9//14

```

题14: sum求和2

```

1  var sum = (function () {
2      var list = [];
3      var add = function () {
4          // 拼接数组
5          var args = Array.prototype.slice.call(arguments);
6          list = list.concat(args);
7          return add;
8      }
9      // 覆盖 toString 方法
10     add.toString = function () {
11         // 计算总和
12         var sum = list.reduce(function (pre, next) {
13             return pre + next;
14         });
15         // 清除记录
16         list.length = 0;
17         return sum;
18     }
19     return add;
20 })();
21 var s = sum(2, 3, 4)(3)(1, 2);
22 console.log(s.toString()); //15
23 console.log(sum(2, 3, 4).toString()); //9
24 console.log(sum(2)(3)(4).toString()); //9
25 console.log(sum(2)(3)(4).toString()); //9
26 console.log(sum(1, 2)(3, 5)(4) / 1) //15

```

题15: 获取字符串的字节长度，中文为2个字节


```
1 var str = "非jnuhii";
2 function getStr(str) {
3     var num = str.length;
4     for (var i = 0; i < str.length; i++) {
5         /*字符串的charCodeAt()方法获取指定索引对应的ASCII码值，汉字的ASCII大于255,其它的字母数字以及
        其他字符ASCII编码值在0-255之间*/
6         if (str.charCodeAt(i) > 255) {
7             num += 1
8         }
9     }
10    return num;
11 }
12 console.log(getStr(str));
```